算法总结

Jiayang Sun

January 2026

1 算法复杂度分析——渐近表示法

大Θ记号 对于一个给定函数$g(n)$，$\Theta(g(n))$表示一个函数的集合：

$$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2, n_0 > 0, \text{s.t.} \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

一般记$f(n) \in \Theta(g(n))$为$f(n) = \Theta(g(n))$。下面的记号同理。

大O记号 对于一个给定函数$g(n)$，$O(g(n))$表示一个函数的集合：

$$O(g(n)) = \{f(n) \mid \exists c, n_0 > 0, \text{s.t.} \forall n \geq n_0, 0 \leq f(n) \leq cg(n)\}$$

大Ω记号 对于一个给定函数$g(n)$，$\Omega(g(n))$表示一个函数的集合：

$$\Omega(g(n)) = \{f(n) \mid \exists c, n_0 > 0, \text{s.t.} \forall n \geq n_0, 0 \leq cg(n) \leq f(n)\}$$

小o记号 对于给定函数$g(n)$，$o(g(n))$表示一个函数的集合：

$$o(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0 > 0, \text{s.t.} \forall n \geq n_0, 0 \leq f(n) < cg(n)\}$$

小ω记号 对于一个给定函数$g(n)$，$\omega(g(n))$表示一个函数的集合：

$$\omega(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0 > 0, \text{s.t.} \forall n \geq n_0, 0 \leq cg(n) \leq f(n)\}$$

证明时只需要设定c和n_0，然后找出对应的值即可。

2 分治法

2.1 解题思路

- 如何分解子问题（Divide）
- 如何递归地解决子问题（Conquer）
- 如何合并子问题的解（Combine）

2.2 复杂度分析

递归树法 根据递归式画出递归树，计算树的所有节点数即为算法的时间复杂度。

主定理 对于形如$T(n) = aT(n/b) + f(n)$的递归式，其中$a \geq 1, b > 1, f(n)$渐近正，记$g(n) = n^{\log_b a}$，其时间复杂度满足：

- 若$g(n)$较大（$f(n) = O(g(n))$），则$T(n) = \Theta(g(n)) = \Theta(n^{\log_b a})$
- 若$f(n)$较大（$f(n) = \Omega(g(n))$），则$T(n) = \Theta(f(n))$
- 若$f(n)$和$g(n)$相当（$f(n) = \Theta(g(n))$），则$T(n) = \Theta(g(n)\log n) = \Theta(n^{\log_b a}\log n)$

注：不是对所有的递归式均满足。例如以下递归式：

$$T(n) = 2T(n/2) + n\log n$$

其中$n^{\log_b a} = n < f(n) = n\log n$，但$f(n)$并不大于$n$一个多项式因子$n^\epsilon, \epsilon > 0$。

对于给定$\epsilon > 0$，对于足够大的n，$n^\epsilon > \log n$，因此不能用主定理求解时间复杂度。

2.3 例题

斐波那契数列

$$F(n) = \begin{cases} 1 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases}$$

全排列问题 设计一个递归算法生成n个元素$\{r_1, r_2, ..., r_n\}$的全排列。

设$R = \{r_1, r_2, ..., r_n\}$为要进行排列的$n$个元素，$R_i = R - \{r_i\}$。集合$X$中的元素的全排列记为$\mathrm{perm}(X)$，$\mathrm{perm}(X)(r_i)$表示在全排列$\mathrm{perm}(X)$的每一个排列后加上后缀$r_i$得到的排列。则$R$的全排列可以归纳定义为：

- 当$n = 1$时，$\mathrm{perm}(R) = (r)$，其中r是集合R中唯一一个元素

- 当$n > 1$时，$\text{perm}(R) = \bigcup_{i=1}^{n} \text{perm}(R_i)(r_i)$

时间复杂度为$O(n!)$。

整数划分 给定一个正整数n，$n = n_1 + n_2 + \cdots + n_k$表示正整数$n$的一个$k$划分，其中$n_1 \geq n_2 \geq \cdots \geq n_k \geq 1$。求正整数$n$的不同划分的个数。

记最大加数$n_1 \leq m$的划分的个数为$q(n, m)$，有

- 当最大加数n_1不大于1时，任何正整数只有一种划分方式：$n = 1 + 1 + \cdots 1$，即$q(n, 1) = 1, n \geq 1$
- 最大加数n_1不能大于n，即$q(n, m) = q(n, n), m \geq n$
- 正整数n的划分由$n_1 = n$的划分和$n_1 \leq n - 1$的划分组成，即$q(n, n) = 1 + q(n, n - 1)$
- 正整数n的最大加数n_1不大于m的划分由$n_1 = m$的划分和$n_1 \leq m-1$的划分组成，即$q(n, m) = q(n, m - 1) + q(n - m, m), n > m > 1$，其中$q(n - m, m)$表示先拿出一个$m$作为分划的一部分，然后再考虑剩下的最大分划数不超过m的分划个数$q(n - m, m)$

因此有递归关系：

$$q(n, m) = \begin{cases} 1 & n = 1, m = 1 \\ q(n, n) & n < m \\ 1 + q(n, n - 1) & n = m \\ q(n, m - 1) + q(n - m, m) & n > m > 1 \end{cases}$$

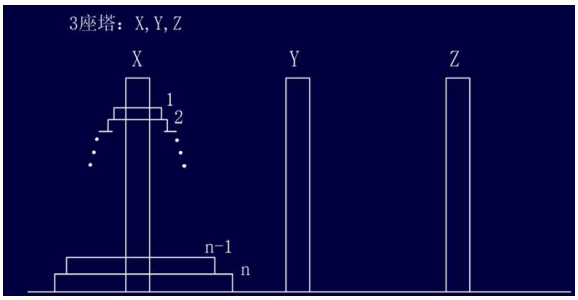

图 1: n阶汉诺塔示意图

n阶汉诺塔问题 将X上的n个圆盘移动到Z上需要多少步？

分解问题：对于第n个盘：

- 将上面$n - 1$个盘先移动到Y上，以Z作为辅助盘（若干步）
- 将第n个盘移动到Z上（一步）

- 将上面的$n-1$个盘移动到Z上，以X为辅助盘（若干步）

当$n=1$时，直接将该盘子从X移动到Z（一步）。

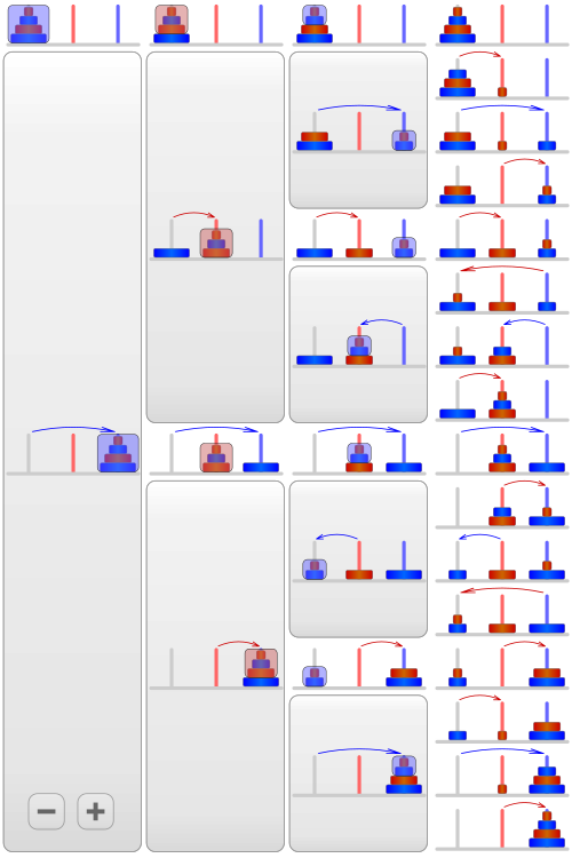

图 2: $n=6$时的解法步骤

思考：对于4个柱子应该怎么做？对于m个柱子（$m<n$）呢？[1]

排序问题　归并排序和快速排序均是基于分治法的排序方法。

1. **归并排序**
2. **快速排序**

二分搜索　适用于有序表

快速傅里叶变换（FFT）　适用于多项式乘法、大整数乘法（整数可以用一个多项式表示）问题。两个$n-1$次的多项式的乘积有$2n-1$个系数。

多项式的点值表示法：记$n-1$次多项式$A(x)=\sum_{j=0}^{n-1}a_jx^j$。则$A(x)$可表示为$n$个点值对$\{(x_k,y_k)\}_{k=0}^{n-1}$，其中$y_k=A(x_k)$。

对应有如下方程组：

$$\begin{pmatrix}1 & x_0 & x_0^2 & \cdots & x_0^{n-1}\\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1}\end{pmatrix}\begin{pmatrix}a_0\\ a_1\\ \vdots\\ a_{n-1}\end{pmatrix}=\begin{pmatrix}y_0\\ y_1\\ \vdots\\ y_{n-1}\end{pmatrix} \tag{1}$$

其中左边为范德蒙矩阵[2]，可记为$V(x_0,x_1,\cdots,x_{n-1})$。

点值表示法的优点：记$A(x)$由$\{(x_k,y_k^A)\}_{k=0}^{n-1}$表示，$B(x)$由$\{(x_k,y_k^B)\}_{k=0}^{n-1}$表示，则$A(x)\times B(x)$在$\{x_0,x_1,...,x_{n-1}\}$处的值为$\{y_0^Ay_0^B,y_1^Ay_1^B,...,y_{n-1}^Ay_{n-1}^B\}$。

FFT的思路：对于两个多项式$A(x)$和$B(x)$，首先将两个多项式转换为点值表示，然后将对应的点相乘，最后使用逆变换将相乘后的点值恢复成多项式。

$$\begin{array}{ccc}A(x),B(x) & \xrightarrow{A(x)\times B(x):O(n^2)} & C(x)\\ \Downarrow \text{转化} & & \Uparrow \text{恢复}\\ \{(x_k,A(x_k))\}_{k=0}^{2n-1},\{(x_k,B(x_k))\}_{k=0}^{2n-1} & \xrightarrow{\text{点值对相乘:}O(n)} & \{(x_k,C(x_k))\}_{k=0}^{2n-1}\end{array} \tag{2}$$

目标：找到一组合适的点对$\{(x_k,y_k)\}_{k=0}^{n-1}$使得方程组(1)有且仅有唯一解。$\Rightarrow$找到一组相异的$x_k$：$x_k$相异 $\iff$ $V(x_0,x_1,\cdots,x_{n-1})$可逆。

目的：可以根据点对恢复出原本的多项式。

考虑如何将多项式高效转换为点值表示：

观察一个$n-1$次多项式

$$A(x)=a_0x^0+a_1x^1+a_2x^2+a_3x^3+\cdots+a_{n-1}x^{n-1}$$

Naïve的做法：直接将n个x带入求解。计算一次$A(x)$需要$O(n)$时间，因此总共的时间复杂度为$O(n^2)$。

原式可以分为奇数项系数和偶数项系数：

$$\begin{aligned}A(x)&=(a_0x^0+a_2x^2+\cdots+a_{n-2}x^{n-2})+(a_1x^1+a_3x^3+\cdots+a_{n-1}x^{n-1})\\&=(a_0x^0+a_2x^2+\cdots+a_{n-2}x^{n-2})+x(a_1x^0+a_3x^2+\cdots+a_{n-1}x^{n-2})\\&=(a_0(x^2)^0+a_2(x^2)^1+\cdots+a_{n-2}(x^2)^{n/2-1})+x(a_1(x^2)^0+a_3(x^2)^1+\cdots+a_{n-1}(x^2)^{n/2-1})\end{aligned}$$

记$A_{even}(x)=a_0+a_2x+a_4x^2+\cdots a_{n-2}x^{n/2-1}$，$A_{odd}(x)=a_1+a_3x+a_5x^2+\cdots a_{n-2}x^{n/2-1}$，则有

$$A(x)=A_{even}(x^2)+xA_{odd}(x^2) \tag{3}$$

可以使用递归的方式先解出$A_{even}(x^2)$和$A_{odd}(x^2)$然后再合并得到$A(x)$。要得到n个点对，可以取任意n个不同的x带入，这样有递归式$T(n) = 2T(n) + O(n)$，根据主定理，复杂度仍为$O(n^2)$。有没有什么方法能取$n/2$个不同的x带入得到n个点对？

观察式(3)可以得到对应$-x$的公式：

$$A(-x) = A_{even}(x^2) - xA_{odd}(x^2) \tag{4}$$

to finish the discription

Strassen矩阵乘法 对于两个矩阵A和B，求二者的乘积$C = AB$。

直接计算：

$$c[i,j] = \sum_{k=1}^{n} A[i,k]B[k,j]$$

时间复杂度为$O(n^3)$。

考虑对矩阵进行分块：

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

于是有：

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

将原$n \times n$矩阵乘法转换为8个$n/2 \times n/2$矩阵的乘法，于是有递归式：

$$T(n) = \begin{cases} O(1) & n = 2 \\ 8T(n/2) + O(n^2) & n > 2 \end{cases}$$

根据主定理，时间复杂度仍为$O(n^3)$。

通过一些手段将8个乘法变成7个乘法：记

$$
\begin{aligned}
M_1 &= A_{11}(B_{12} - B_{22}) \\
M_2 &= (A_{11} + A_{12})B_{22} \\
M_3 &= (A_{21} + A_{22})B_{11} \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_6 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
M_7 &= (A_{11} - A_{21})(B_{11} + B_{12})
\end{aligned}
$$

于是有

$$
\begin{aligned}
C_{11} &= M_5 + M_4 - M_2 + M_6 \\
C_{12} &= M_1 + M_2 \\
C_{21} &= M_3 + M_4 \\
C_{22} &= M_5 + M_1 - M_3 - M_7
\end{aligned}
$$

递归式变为

$$
T(n) = \begin{cases} O(1) & n = 2 \\ 7T(n/2) + O(n^2) & n > 2 \end{cases}
$$

时间复杂度为$O(n^{\log 7}) = O(n^{2.81})$。

平面最近点对 给定一个二维平面上的n个点，求找出其中距离最小的一对点。

易得暴力方法的时间复杂度：$O(n^2)$。

分治的思路：将平面分为两个子平面，递归找出两个子平面中距离最短的点对，记距离为d_1和d_2，和跨两个子平面的点对的距离对比，得到距离最小的点对。

记$\delta = \min\{d_1, d_2\}$，那么只需要分析分割线左右$\delta$范围即可。

时间复杂度分析：有递归式$T(n) = 2T(n/2)+O(n)$，根据主定理有时间复杂度为$O(n \log n)$。

棋盘覆盖问题 在一个$2^k \times 2^k$个方格组成的棋盘中，恰有一个方格与其它方格不同，称该方格为一特殊方格，且称该棋盘为一特殊棋盘（如图 4）。在棋盘覆盖问题中，要用图示的4种不同形态的L型骨牌覆盖给定的特殊棋盘上除特殊方格以外的所有方格，且任何2个L型骨牌不得重叠覆盖。

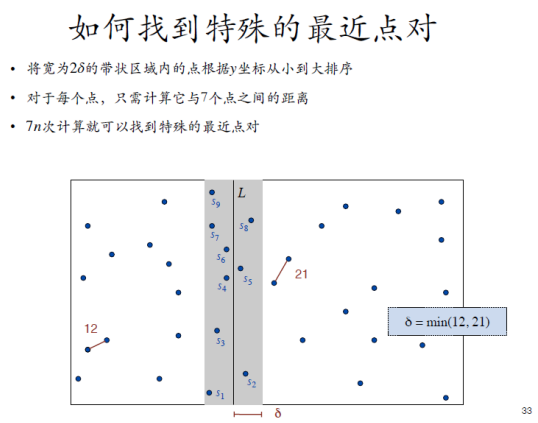

图 3: 如何找到跨分割线的最近点对

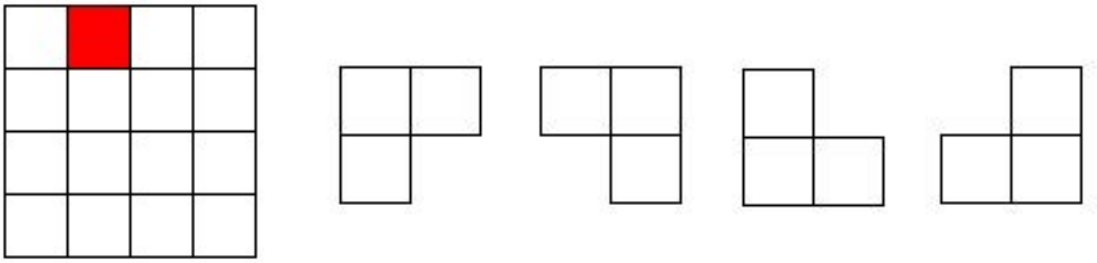

图 4: 棋盘问题示例

可以将$2^k \times 2^k$的棋盘分割为4个$2^{k-1} \times 2^{k-1}$的子棋盘，必有一个子棋盘为特殊棋盘。可以使用一个骨牌将剩下三个子棋盘转化为特殊棋盘，于是将原本问题转化为4个子问题，如图5所示。可得递归式

$$T(k) = 4T(k-1) + O(1)$$

时间复杂度为$O(4^k)$。

循环赛日程表问题　设计一个满足以下要求的日程表：

- 每个选手必须与其他$n-1$个选手各赛一次
- 每个选手一天只能赛一次
- 循环赛一共进行$n-1$天

可将所有选手分为两半，n个选手的比赛日程表可以通过为$n/2$个选手设计的比赛日程表来决定。递归地分割直到只有两个选手，只需要让这两个选手进行比赛即可。

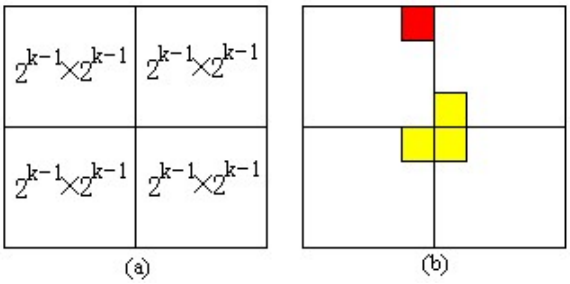

图 5: 递归求解棋盘问题

3 动态规划

3.1 解题思路

- 定义状态
- 构造状态转移（递归关系）
- 自底向上计算

动态规划和分治法的区别：更适合解决有重叠子问题的情况。

3.2 例题

最长公共子序列（LCS） 给定两个长度分别为m和n的序列X和Y，若Z既是X的子序列，又是Y的子序列，则Z为X和Y的公共子序列。要求X和Y的公共子序列中长度最大者。

记$C[i,j]$为X_i和Y_j的LCS长度。则对于$0 \le i \le m, 0 \le j \le n$，有

$$C[i,j] = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ C[i-1,j-1]+1 & i,j > 0, x_i = y_j \\ \max\{C[i,j-1], C[i-1,j]\} & i,j > 0, x_i \neq y_j \end{cases}$$

初始化行和列，然后依次由$i = 1, 2, ..., m$和$j = 1, 2, ..., n$递推计算，最终$C[m,n]$即为所求解。

可通过将C压缩为$2 \times n$的数组求解，将空间复杂度缩小为$O(m+n)$[1]。

0-1背包问题 给定n种物品，其中第i种物品的价值为v_i，重量为w_i。现有一背包，可以装总质量不超过C的若干物品，问如何选择装入背包的物品，使得在不超出背包容量的情况下，装入背包中的物品总价值最大？

[1] $O(n)$?

记$m[i,j]$表示背包容量为j，可选择物品为$0,1,...,i$时的最优解。则有递归式：

$$m[i,j]=\begin{cases}\max\{m[i-1,j],m[i-1,j-w_i]+v_i\} & j\geq w_i\\ m[i-1,j] & 0\leq j<w_i\end{cases} \tag{5}$$

含义为：若当前背包可以装下第i个物品（$j\geq w_i$），则比较装物品i后的总价值（$m[i+1,j-w_i]+v_i$）和不装物品i后的总价值（$m[i+1,j]$）；否则不装物品i（总价值和$m[i+1,j]$的总价值相同）。

也可以使用一维动态规划求解：记$m[j]$表示背包容量为j时的最大总价值。那么对于每一个物品i，有递归式：

$$m[j]=\begin{cases}\max\{m[j],m[j-w_i]+v_i\} & j\geq w_i\\ m[j] & 0\leq j<w_i\end{cases}$$

含义与式(5)相似。注意：这里的j需要从C到1遍历。

4 贪心算法

4.1 例题

活动选择问题 设有n个活动$S=a_1,a_2,...,a_n$，均要使用某资源（如同一间教室），该资源使用方式为独占式，一次只供一个活动使用。每个活动a_i发生的时间为$[s_i,f_i),0\leq s_i<f_i<\infty$。两个活动相容（不冲突），是指其中一个活动的开始时间必须大于等于另一个活动的完成时间。问如何选择，使得相容活动的集合最大？

贪心思路：按照活动结束时间进行顺序排序，然后从小到大依次选择相容活动。

思考：为什么不可以以开始时间最早、活动持续时间最短、与其他活动重叠最少来贪心？

部分（分数）背包问题 给定n种物品，其中第i种物品的价值为v_i，重量为w_i。现有一背包，可以装总质量不超过C的物品，其中每种物品可以取出部分装入背包。问如何选择装入背包的物品，使得在不超出背包容量的情况下，装入背包中的物品总价值最大？

贪心思路：计算出每种物品的单位价值，然后以从大到小的顺序依次装入背包，直到背包被装满。

5 概率算法

5.1 Sherwood算法

确定性算法通常假设算法的输入实例满足某一特定的概率分布。很多算法对不同输入实例运行时间差别很大（例如快速排序算法），可采用Sherwood概率算法消除时间复杂度与输入实例间的依赖关系：

- 在确定性算法的某些步骤引入随机因素
- 对输入实例进行随机处理再进行确定性算法

5.2 Las Vegas算法

方式：重复随机决策直到获得了成功的结果再返回。

特点：要么返回正确的解，要么随机决策导致一个僵局。若陷入僵局使用同一实例运行同一算法有独立的机会求出解。成功的概率随执行时间的增大而增大。

对于实例x，设LV算法单次成功率为$p(x)$，成功时的期望时间为$s(x)$，失败时的期望时间为$e(x)$，则总期望时间为：

$$\begin{aligned} t(x) &= p(x)s(x) + (1-p(x))(e(x)+t(x)) \\ \Rightarrow t(x) &= s(x) + \frac{1-p(x)}{p(x)}e(x) \end{aligned}$$

如何最小化$t(x)$?

5.3 Monte Carlo算法

概念：

- 设p是一个实数，且$1/2 < p < 1$，若一个Monte Carlo算法以不小于p的概率返回一个正确的解，则该MC算法称为p正确，算法的优势为$p - 1/2$
- 若一个MC算法对同一实例绝不给出两个不同的正确解，则该算法称为相容的或一致的

一个MC算法是偏真的，指的是：算法可能把“真”错判为“假”，但绝不会把“假”错判为“真”。

重复调用k次一致、p正确、偏真的MC算法，可得到一个$(1-(1-p)^k)$正确的算法，即假设k次全是错误的结果，那么该算法的正确率为$(1-(1-p)^k)$。

如果要控制出错概率小于$\epsilon > 0$，那么可以重复调用k次，其中

$$(1-p)^k < \epsilon \Rightarrow k < \frac{\log \epsilon}{\log(1-p)}$$

可取$k = \left\lceil \dfrac{\log \epsilon}{\log(1-p)} \right\rceil$

特点：偶尔会出错，但对于任何实例均能以高概率找到正确解。算法运行的次数越多，得到正确解的概率越高。

5.4 例题

数字积分 对于定积分$\int_a^b f(x)dx$，求该积分的值可以在积分区间$[a,b]$上随机均匀地产生点，求这些点上地函数值的算术平均值，然后再乘以区间的宽度：

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{n}\sum_{i=1}^{n} f(x_i), a \le x_i \le b$$

将上式写为

$$\int_a^b f(x)\mathrm{d}x = \sum_{i=1}^{n} f(x_i)\frac{b-a}{n}, a \le x_i \le b$$

等同于根据定积分的定义：

$$\int_a^b f(x)\mathrm{d}x = \lim_{\xi_i \to 0}\sum_{i=1}^{n} f(\xi_i)\Delta x_i$$

进行采样求解。

Algorithm 1 概率算法进行数字积分计算定积分的值

Require: 函数f，采样量n，积分区间a, b

Ensure: 函数f在区间$[a,b]$上的积分

```
sum ← 0
for i ← 1 to n do
    x ← uniform(a, b)
    sum ← sum + f(x)
end for
return (b − a)sum/n
```

求集合的势（大小） 设X为具有n个元素的集合。有放回地随机、均匀和独立地从X中选取元素，设k为出现第1次重复之前所选出的元素数目，则当n足够大时，k的期望趋近于$\beta\sqrt{n}$，其中$\beta = \sqrt{\pi/2} \approx 1.253$[2]

[2] 如何证明？

于是可以使用以下公式设计估计$|X|$的概率算法：

$$\beta\sqrt{n} = \sqrt{\frac{n\pi}{2}} = k \Rightarrow n = \frac{2k^2}{\pi}$$

```
QUEENS_LV(success) //若success=true，则try[1..8]包含8后问题的一个解
1   col, diag45, diag135 ← ∅      //冲突列及两斜线集合初始为空
2   k ← 0    //行号
3   repeat    //try[1..k]是k-promising，考虑放第k+1个皇后
4       count ← 0   //计数器，count值为第k+1个皇后的安全位置总数
5       for i ← 1 to 8 do    //i是列号，试探(k+1, i)是否安全
6           if i ∉ col and i – k – 1 ∉ diag45 and i + k + 1 ∉ diag135
7               count ← count + 1
8               if RANDOM(1, count) = 1  //以1/count概率在count个安全位置上
9                   j ← i                //随机选择1个位置j放置皇后
10      if count > 0        //count=0时无安全位置，第k个皇后尚未放好
11          k ← k + 1       //try[1..k+1]是(k+1)-promising
12          try[k] ← j
13          col ← col ∪ {j}
14          diag45 ← diag45 ∪ {j – k}
15          diag135 ← diag135 ∪ {j + k}
16  until count = 0 or k = 8   //当前皇后位置搜索失败或8-promising时结束
17  success ← (count > 0)
```

图 6: 八皇后问题的LV算法伪代码

八皇后问题

主元素问题　设$A[1..n]$是含有n个元素的数组，若A中等于x的元素个数大于$n/2$，则称x是数组A的主元素。

MC算法：随机从A中选一个元素，然后计算这个元素的个数，并判断是否是主元素。是一个偏真1/2正确的算法。

矩阵乘法验证　设A, B, C为$n \times n$矩阵，如何判定$AB = C$是否正确？

MC算法：构造一个随机0/1二值行向量，将判断$AB = C$改为判断$XAB = XC$。时间复杂度为$O(n^2)$。是一个偏假的正确概率为1/2的MC算法，因为当返回false时算法一定正确。

6 分布式算法

6.1 模型

考虑：

- 异步消息传递模型：用于松散耦合机器及广域网
- 同步消息传递模型：一个理想的消息传递系统。该系统中，某些计时信息（如消息延迟上界）是已知的，系统的执行划分为轮执行，是异步系统的一种特例。

6.2 基本算法

6.2.1 形式化模型（或许可以通过戴森球的一个产线来理解）

拓扑 无向图，其中节点表示处理机，边为双向信道。

算法 由系统中每个处理器上的局部程序构成：一个系统或一个算法是由n个处理器$p_0, p_1, ..., p_{n-1}$构成，每个处理器p_i可以模型化为一个具有状态集Q_i的状态机（可能是无限的）。

局部程序 负责执行局部计算，以及发送和接收消息。

状态（进程的局部状态） 由p_i的变量，p_i的msgs构成。p_i的每个状态由$2r$个msg集构成：

- $\text{outbuf}_i[l], 1 \leq l \leq r$：$p_i$经第$l$条关联的信道发送给邻居，但尚未传到邻居的msg
- $\text{inbuf}_i[l], 1 \leq l \leq r$：在$p_i$的第$l$条信道上已传递到$p_i$，但尚未经$p_i$内部计算步骤处理的msg。

初始状态 Q_i包含一个特殊的初始状态子集，其中每个$\text{inbuf}_i[l]$必须为空，但$\text{outbuf}_i[l]$未必为空。

转换函数（transition） 处理器p_i的转换函数为：

- 输入：p_i可访问的状态
- 输出：对每个信道l，至多产生一个msg输出
- 转换函数使$\text{inbuf}_i[l], 1 \leq l \leq r$清空

配置 配置是分布式系统在某点上整个算法的全局状态，表示为向量$(q_0, q_1, ..., q_{n-1})$，q_i是p_i的一个状态。一个配置里的outbuf变量的状态表示在通信信道上传输的信息，由del(传递)事件模拟传输。一个初始的配置为向量$(q_0, q_1, ..., q_{n-1})$，其中每个q_i是p_i的初始状态，即每个处理器处于初始状态。

事件　系统里所发生的事情均称为事件，对于msg传递系统，有：

- comp(i)：计算事件，代表处理器p_i的一个计算步骤。其中p_i的转换函数被用于当前可访问的状态
- del(i, j, m)：传递事件，表示msg m从p_i传送到p_j

执行　系统在时间上的行为被称为一个执行。是一个由配置和事件交错构成的序列。该序列需主要满足以下两类条件：

- Safety条件（安全性），表示某个性质在每次执行中每个可到达的配置里都必须成立。在序列的每个有限前缀里必须成立的条件
- Liveness条件（活跃性），表示某个性质在每次执行中的某些可达配置里必须成立。必须成立一定次数的条件（可能是无数次）

对特定系统，满足所有要求的安全性条件的序列称为一个**执行**；若一个执行也满足所有要求的活跃性条件，则称为**容许（合法）执行**。

6.2.2　同步系统和异步系统

参考文献

[1] Menon A. On The Optimal General Solution To The Multi-Peg Tower of Hanoi[EB/OL]. 2025. https://arxiv.org/abs/2505.12941. arXiv: 2505.12941 [math.CO].

[2] 维基百科贡献者. 范德蒙德矩阵[EB/OL]. 2025. https://zh.wikipedia.org/w/index.php?title=%E8%8C%83%E5%BE%B7%E8%92%99%E5%BE%B7%E7%9F%A9%E9%98%B5&oldid=90577510.